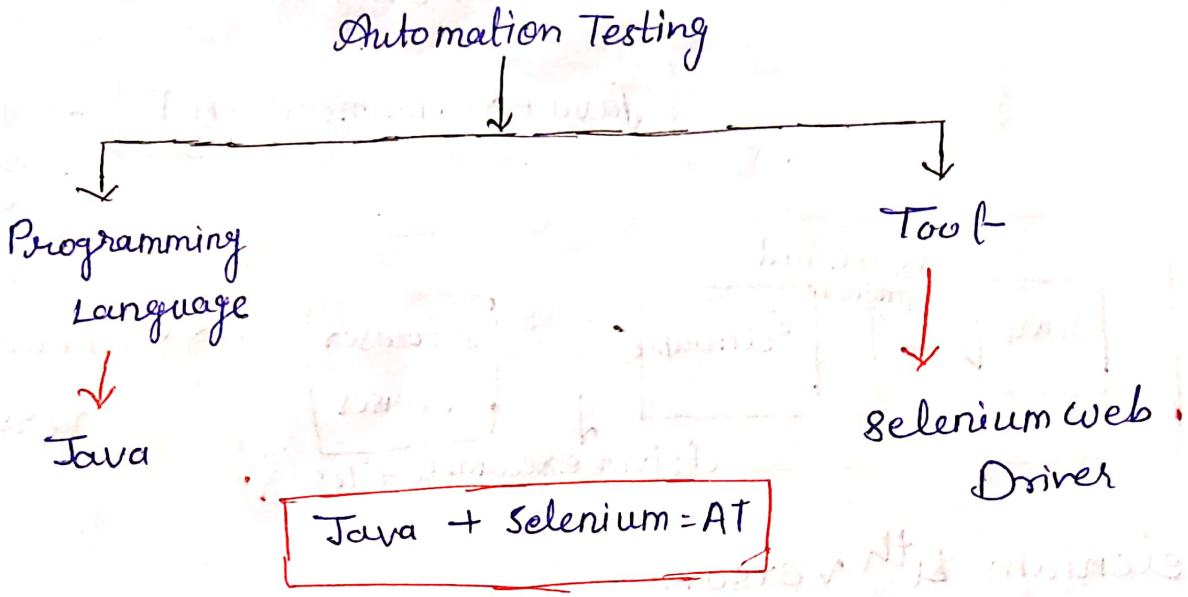
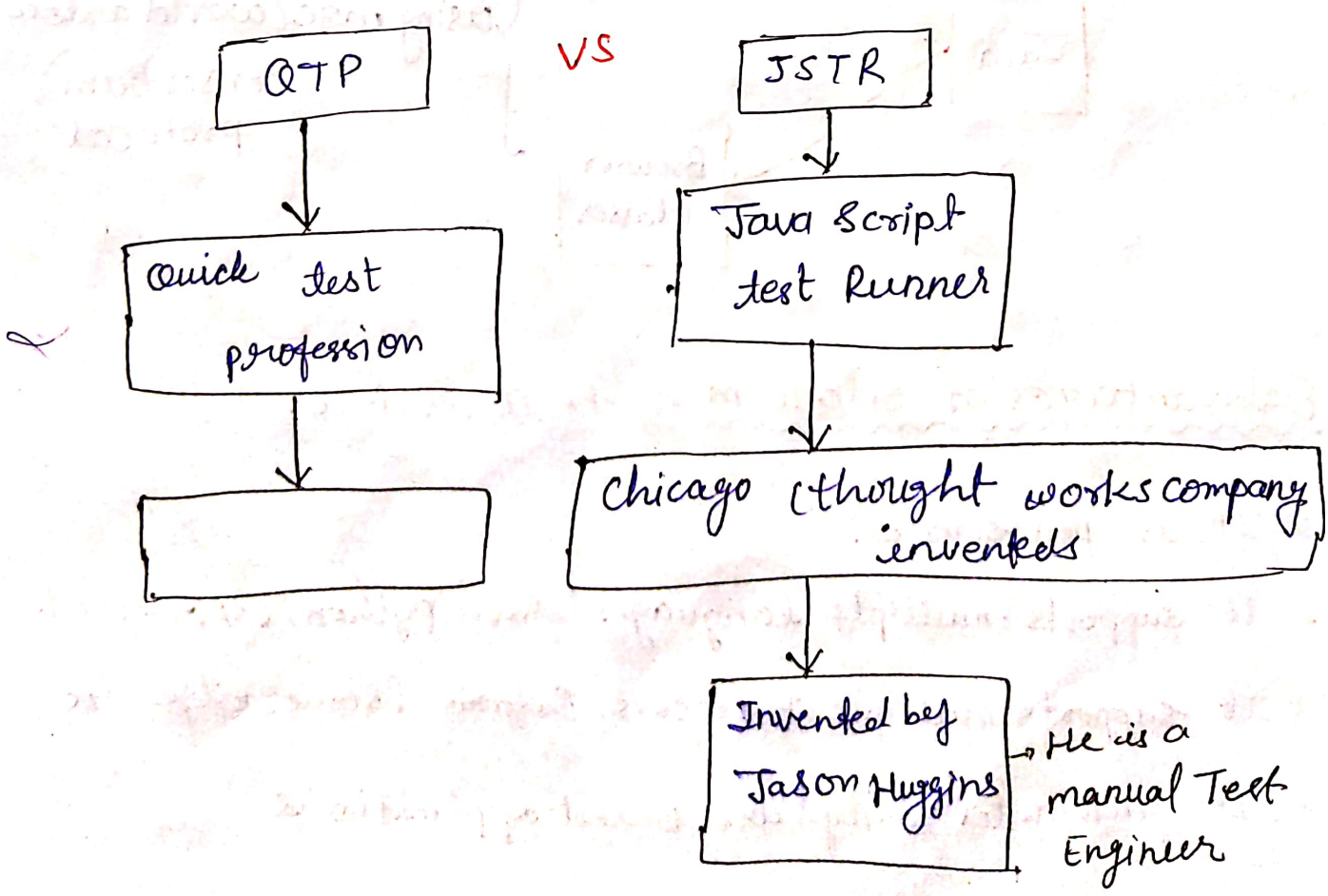
Automation Testing

The process of converting manual Testcases into automation script is called as "Automation Testing."



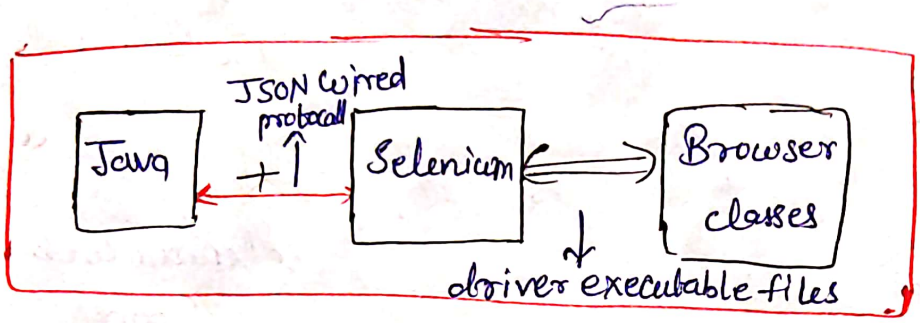
Selenium (injection



[16/09/24]

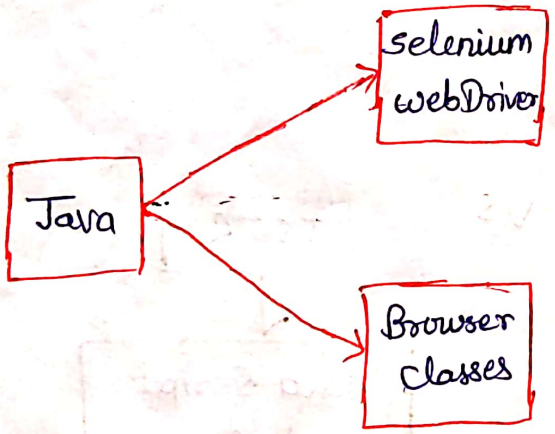
Selenium WebDriver :- It is the tool which is used to automate web Applications (The applications which will open through "URL" / Links)

Java + Seleniumwd = AT → Java Selenium wD



⇒ Selenium 3rd version

Selenium 4th version



using w3c (world wide web consortium) protocol

Advantages of Selenium :- 1. It is free.

2. It is open source.

3. It supports multiple languages Java, Python, C#, .Net etc.

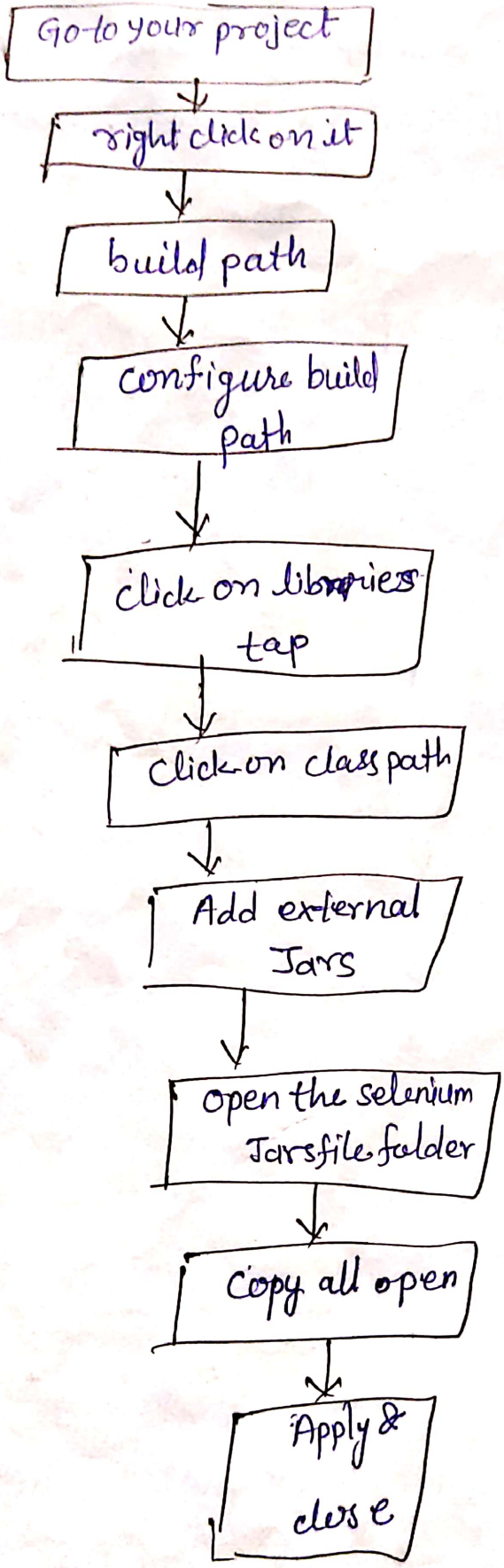
4. It supports multiple Browsers Safari, Chrome, edge etc.

5. It automates only web based applications.

Disadvatages of Selenium:-

1. It won't automate desktop based app's
2. It won't automate captcha, barcode, videos ~~etc~~ QTP's etc.

How to add seleniumJars to Eclipse?



Create Java project in Eclipse with the name Basic Selenium.

WebDriver Methods :- It is interface which have 13 abstract methods which are useful for Automation program.

get() :- It is used to open application / It is used to enter Url.

```
WD driver = new ChromeDriver();
driver.get("Url");
```

Navigate() :- It is also used to open the application, and move forward, Backward and refresh the browser. It is alternative method for get().

Ex.-

```
WebDriver driver = new ChromeDriver();
driver.navigate().to("https://www.google.com");
Thread.sleep(4000);
driver.navigate().to("https://www.amazon.in");
driver.navigate().back();
driver.navigate().forward();
driver.navigate().refresh();
Thread.sleep(4000);
}
}
```

Methods in WebDriver Interface

#getUrl →

Thread.sleep():- It is type of wait which comes from Java.

→ It will wait for particular interval of time and then perform the next scenario.

→ Here time is always given in milliseconds

→ Thread.sleep(milliseconds)

③ close():- It will close the current tab.

driver.

④ quit():- It is the alternative method for close. It will close all the tabs.

driver.

⑤ getcurrenturl():- It is used to set Url of current webpage. Returntype of getCurrentUrl is string.

```
driver.get("www.google.com");
String exp url = "";
```

6) getTitle() :-

7) getpage source() :- Right click on webpage
↓
view page source
↓
will see entire sourcecode.

```
String pgsrc = driver.setpage source();
System.out.println(pgsrc);
```

8) Setsize() :- It set the is used to change the size of Browser. which accept Dimenssion class arguments.

```
WebDriver driver = new ChromeDriver();
driver.get("Url");
Dimension d = new Dimension(100, 200);   // height, width
driver.manage().window().setsize(d);
```

9) setposition() :- It used to change the position of browser which accepts point class arguments.

```
WebDriver driver = new ChromeDriver();
driver.get("Url");
Point p = new point(int x, int y) - cordinators
driver.manage().window().Setposition(p);
```

(10) Maximize() :- It is used to maximize webpage.

```
driver.manage().window().maximize();
```

(11) Minimize() :- It is used to minimize webpage.

[3.141592 it is not there]

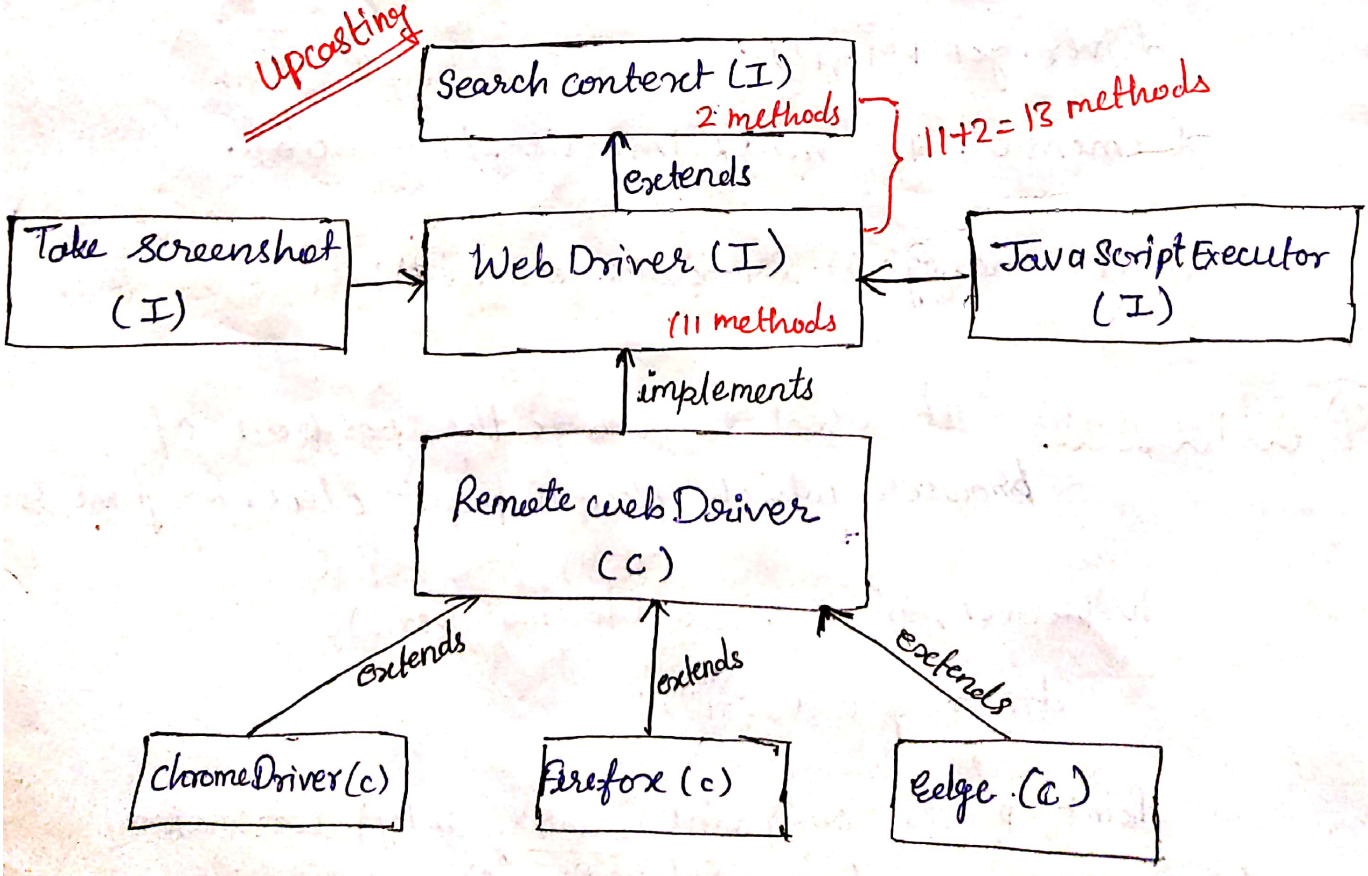
```
driver.manage().window().minimize();
```

Remaining Methods :- 1. SwitchTo()
2. getWindowHandle()
3. findElement()
4. findElements()

WebDriver Architecture :- It contains two Interfaces

→ SearchContext(I)
→ WebDriver (I)

1 class → RemoteWebDriver (class)



Search Context (I) :- It contains 2 methods [findElements(), findElement()]

and WebDriver contains 11 methods. Due to inheritance 2 methods from SearchContext(I) is added to WebDriver(I) total methods in WebDriver(I) are 13 abstracts.

To complete those abstract methods we have implementation class (RemoteWD).

Why we are writing WD d = new CD(); ?

⇒ To access 13 abstract methods from WD(I).

⇒ For Runtime polymorphism.

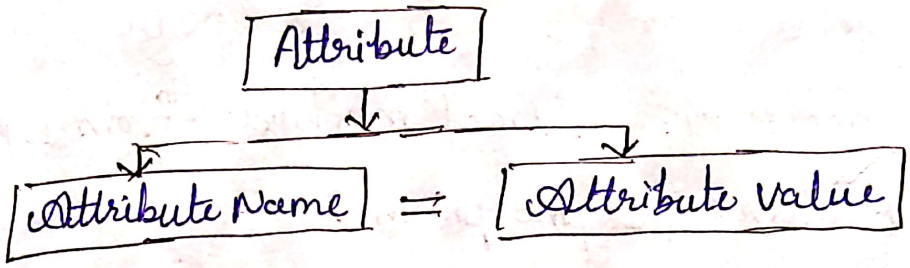
[19/09/24]

HTML Basics:- (Hypertext markup Language)

① tagname:- Anything which is present immediately after "<" is called as "tagname"

ex - <input ; <a , <div , <html , <span id='123'

② Attribute :-



ex - id = "1234" → AV always should written in single or double quote.

③ Visible Text:-

ex - >gmail< , >Images<

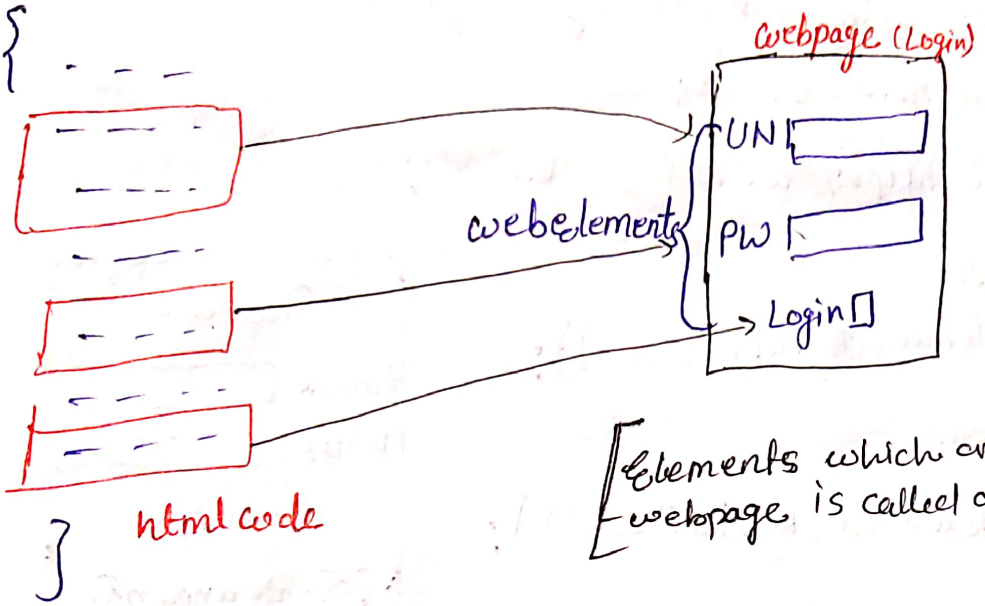
NOTE → Always it is in Black color.

★ tagnames :-

<textarea

1. <body
2. <div
3. <script
4. <link
5. <iframe
6. <img
7. <a
8. <nav
9. <header
10. <span
11. <input
12. <from
13. <footer
14. ~~<Aink~~
15. <ul
16. <li
17. <svg
18. <label
19. <select

Locators :- These are helping to find the web-element inside the html code.



Elements which are present on webpage is called as webelements

⇒ Locators are helping to find element / find elements methods to find web element inside the html code.

⇒ Locators are static methods present in "By" class.

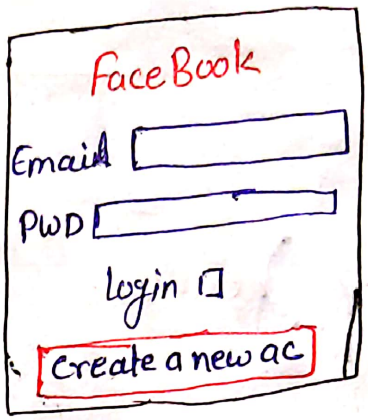
⇒ There are 8 locators.

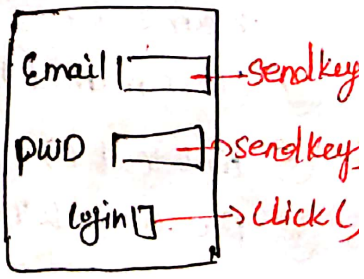
1. tagname() ⟶ faster but not unique
2. id() ⟶ fast and unique
3. name()
4. class name()
5. linkText()
6. partial linkText()
7. css-selector()
8. x-path() ⟶ slow and useful

new locator → relativepath()

How Locator will work?

```
public class FB{
public static void main (String [] args) {
  WD d = new CD ();
    d.get ("https:// www.facebook.com");
   // Email
driver.findElement(By.id ("___"));
   // Password
driver.findElement (By.id ("___"));
   // login
driver.findElement (By.id ("___"));
   // create a new account
driver.findElement (By.id ("___"));
  }
  }
```



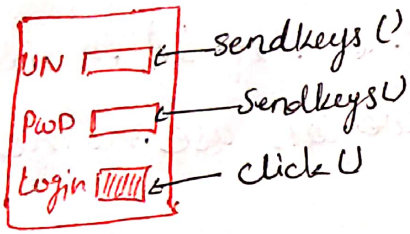


WebElement(I)

- textboxes → accepts values → sendkeys()
- link() → click → click()
- Button → click → click()
- CheckBoxes → ~~select~~ select/click isselected()
- RadioButton → click (select is selected()
- Hyperlinks → click → click()

```
// email
driver.findElement(By.id("___")).sendKeys("admin"));
// password
driver.findElement(By.id("___")).sendKeys("pass@123"));
// login
driver.findElement(By.id("___")).click();
```

findElements — performing methods



since it is an interface all the methods are abstracts methods only

① Sendkeys() → This method is used to enter the value in the input field / Text field

② Click() → This method is used to perform the value click on Buttons, links, Radio Buttons and checkbox etc.

③ Clear() → It is used to clear value present in the text field / textbox

④ isEnabled() → It is used to verify element is enabled or disabled. Return type is boolean.

Selenium 4 Locators :- [20/9/24]

① tagname() :- It is faster locator but not unique so we can't use it.

ex - // input ⇒ It will give more than 1 of 1 matching

Syntax :- driver.findElements(By.tagname("—"));

② id() :- It is faster and it is unique. using id we can make 1 of 1 matching

Syntax :- driver.findElement(By.id("AV"));

③ name() :- Whenever id is not there or if id is not unique inside html then we can use name locater.

Syntax :- driver.findElement(By.name("AV"));

④ classname() :- If id, name both locators are not there or if they are not unique then we can use classname locator.

Syntax :- drivers.findElement(By.classname("—"));

⑤ linkText() / visibleText :- whenever link is there then we can go through it. (If, id, name, classname is not there [This case is very rare case] then we can go through linktext locator / visibletext.)

Syntax :- driver.findElement(By.linkText("—"));

[20/9/24]

Scenario:- 1. Locators :-

① tagname() :- It is faster locator but not unique so we can't use it.

ex - // input ⇒ It will give more than 1 of 1 matching

Syntax :- driver.findElements(By.tagname("—"));

② id() :- It is faster and it is unique. using id we can make 1 of 1 matching

Syntax :- driver.findElement(By.id("AV"));

③ name() :- Whenever id is not there or if id is not unique inside html then we can use name locater.

Syntax :- driver.findElement(By.name("AV"));

④ classname() :- If id, name both locators are not there or if they are not unique then we can use classname locator.

Syntax :- drivers.findElement(By.classname("—"));

⑤ linkText() / visibleText :- Whenever link is there then we can go through it. (If, id, name, classname is not there [This case is very rare case] then we can go through linktext locator / visibletext.)

Syntax :- driver.findElement(By.linkText("—"));

⑥ partialLinkText();- Instead of taking entire text we can take only few letters from partial text

Syntax:- driver.findElement(By.partialLinkText("GM"));

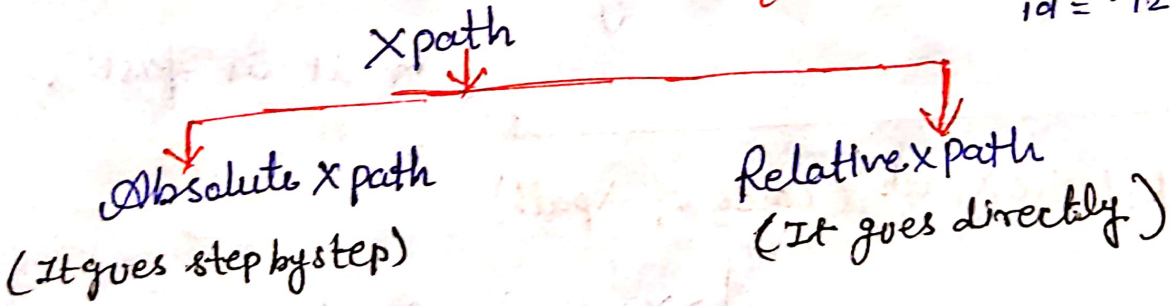
⑦ CSS-Selector:- It will be applicable for every attribute

ex:- id = '123'
name = 'abc'
class name = 'xyz'
title = "Submit"
placeholder = "Sample"

Syntax:- driver.findElement(By.css-selector("—"));

⑧ "X-path":- (imp) It is the path travelled inside the html code.
There are 2 types of X-path

[Syntax:- driver.findElement(By.xpath("//input[@id='1234']"));]



Why we are using Xpath as we have these many locators?

⇒ It is having multiple cases so that we are using xpath [specially Relative xpath].

Why Relative

⇒ Because it is not lengthy and it is not time consuming.

Difference between absolute and Relative Xpath:

<html ⇒ root of parent:
<a immidiat child
<b
<c
<d
<e ⇒ any child

ⓐ Absolutexpath
html/a/b/c/d/e

ⓑ Relative Xpath
html//e

Absolutexpath	Relative xpath
It will navigate from root of parent to immidiate child.	→ It will navigate from root of parent to any child.
It is denoted by "/".	→ It is denoted by "//".
It is lengthy.	→ It is not lengthy.
It is time consuming	→ It is not time consuming or it is fast.

Relative Xpath (Cases of Xpath)

- → Xpath By Attribute
- → Xpath By Text
- → Xpath By contains
- → Xpath By Index

Case-1 Xpath By attribute - It is applicable for any attribute.

<input id = 'abc'

formula - //tagname[@AN ='AV']

⇒ Shortest distance(//)

⇒ time less

⇒ Script is short

Case-2 Xpath By visible Text :- >text<

formula :- //tagname[text() = 'textvalue']

>gmail<

//input[text() = 'gmail']

Case-3 xpath By groupIndex :- (xpath)[ ] → Indexnumber

It converts multiple matchings to single matching.

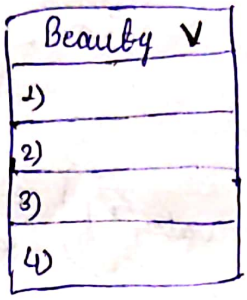
If we want to convert multiple matching to single matching ie 1 of 1 then we have to go through Xpath By Group index.

(xpath Expression)[index]

Case-4 : Xpath By Contains :-

[28/09/24]

Dropdown / Listbox :- It is the set of elements from where we can select any one element from it is called as "Dropdown / List Box".



Dropdown

Static Dropdown [Created using Select tagname]

Dynamic/Autos

[Created without using select tagname]

Singleselectable

Multiselectable

How to create single-selectable DD?

```
<html>
<body>
<select id = 'Good Hotel.'>
<option value = 'a'> Idly </option>
<option value = 'b'> Dosa </option>
:
</select>
</body>
</html>
```

[Note: Save this file with .HTML extension]

How to Handle DropDown

Step-1 store the DropDown in the reference variable

Step-2 Create object of select class which accepts webelement argument.

Step-3 use select class methods to select options.

Ex:- Methods in select class :-

A) Selection methods :- 1) selectByVisibleText()
2) selectByValue()
3) selectByIndex()

B) Deselection methods :- 1) deselectByIndex()
2) deselectByValue()
3) deselectByVisibleText()
4) deselectByAll()

```
public class Day_DropDown {
        main(String[] args){

WebDriver driver = new ChromeDriver();

    driver.get("https://www.facebook.com");

// create new account (click)

driver.findElement(By.xpath("__")).click();

// write xpath for DD and store in reference variable

    WebElement day = driver.findElement(By.xpath("__"))

// Create object of select class

Select s = new Select(day);
  s.selectByVisibleText("15");
}
```

// select month

```
WebElement month = driver.findElement(By.xpath("..."));
    Select s1 = new Select(month);
        s1.SelectByVisibleText("Feb");
```

// select year

```
Thread.Sleep(3000)
WebElement year = driver.findElement(By.xpath("..."));
    Select s2 = new Select(year);
        s2.SelectByVisibleText("1994");
```

// handle Radio Button

T

```
WebElement rd = driver.findElement(By.xpath("..."));
                                        click();
```

driver.

How many elements are present in DD?

```
List <webElement> alloption = day.findElements (By.tagname("li"));
    system.out.println (alloptions.size); // 31
```

// print all dates

```
for (int i =0; i<alloptions.size(); i++) {
    system.out.println (i.getText());
    }
```

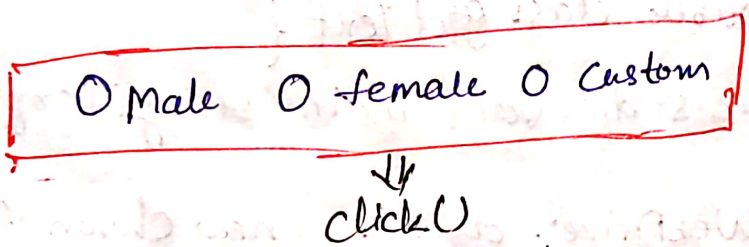
⇒ getText() → It is Method present in webelement (I) which returns the text of webelement.

```
public class GetText {
public static void main (String[] args) {
    WebDriver driver =new ChromeDriver();
        driver.get ("google");
String t =driver.findElement(By.linkText("ABOUT")).getTex
                                                    ();
    system.out.println (t);
    }
    }
```

* Difference b/w findElement() and findElements()

findElement()	findElements()
It will find single matching element.	It will find multiple matching elements.
Returntype of findElement is WebElement	Returntype of findElements is List<WebElement> ↳ Iterface present in collection

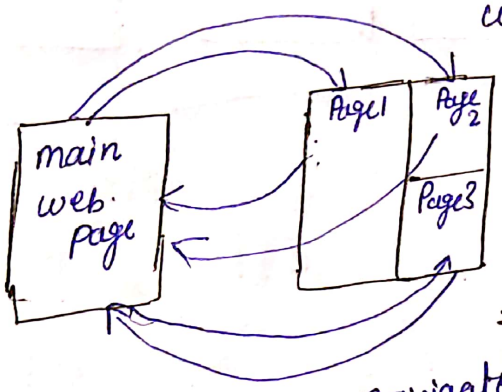
It is set of element from which user can click only one called "Radio Button"



we will use isSelected() for varification.

```
public class RadioBtn {
public static void main(String[] args){
WebDriver driver = new ChromeDriver();
driver.get("Url");
// select male
WebElement male = driver.findElement(By.xpath(xpath Expration));
male.click()
if(male.isSelected()){
System.("RadioBtn is selected")
} else {
Syso.
```

I-frame → One webpage can be represented as multiple webpage, individual webpage is called as "frame/iframe".



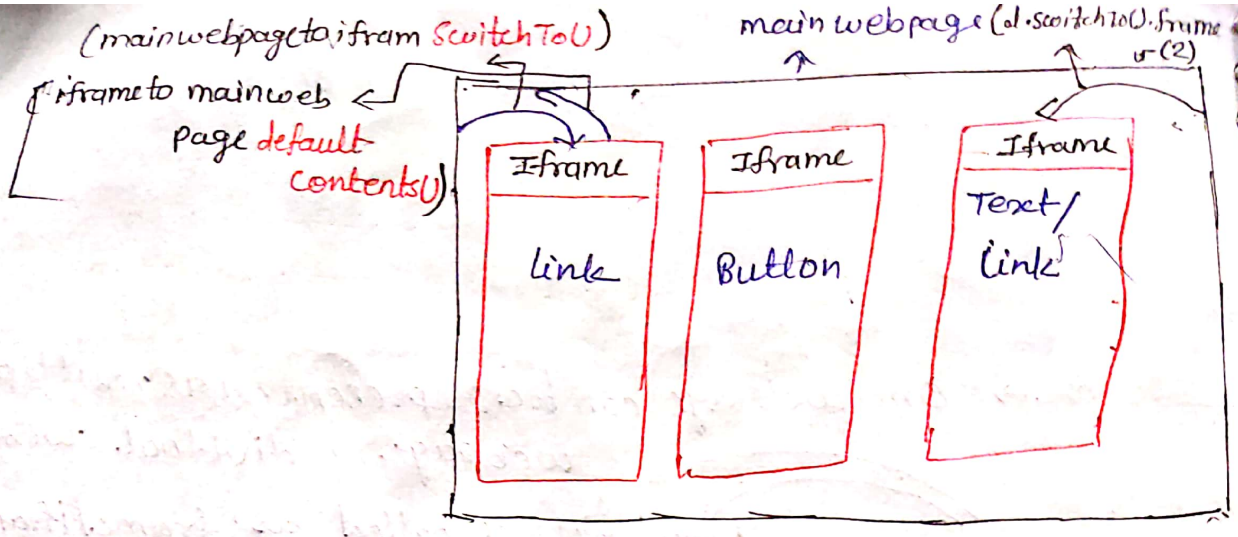
⇒ switchTo() → It will navigate to fram.

⇒ defaultcontent() → It will navigate from any child/frame to default / main web page.

⇒ ParentFram() ✗

```
public . class{
public static void main (String [] args){
    WebDriver driver = new ChromeDriver();
    driver.get ("Url");

    // Swith to frame
    driver.switchTo().frame(2);   // 2 → frameindex
    // Click on The Rock says
    driver.findElement (By.xpath ("____")).click();   // ____ → xpath expression

    }

    }
```

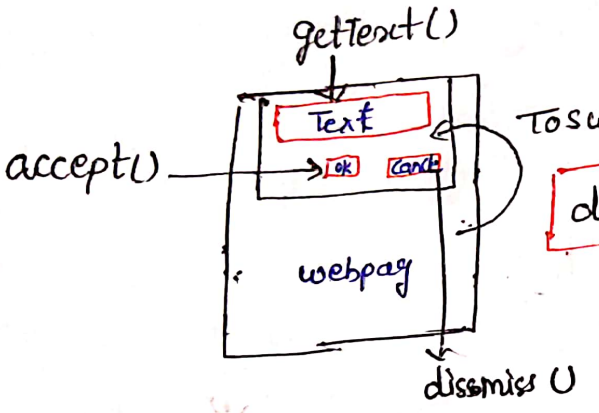
NOTE → If we take any other locator rather than id/name then we will get "No such frame Exception"


(main webpage to iframe SwitchTo())
main webpage (dr.switchTo().frame or (2))
iframe to main web page default contents()
Iframe
link
Iframe
Button
Iframe
Text/
Link

Popups:- These are the small windows which will displayed on webpage while performing action on it. [25/09/24]

Characterstics:-

1. It is not colourful
2. It is not inspectable
3. It is not draggable
4. It contains ok, cancle, !, ? text.
5. We can't move ahead without handeling it.



To switch control from webpage to popup

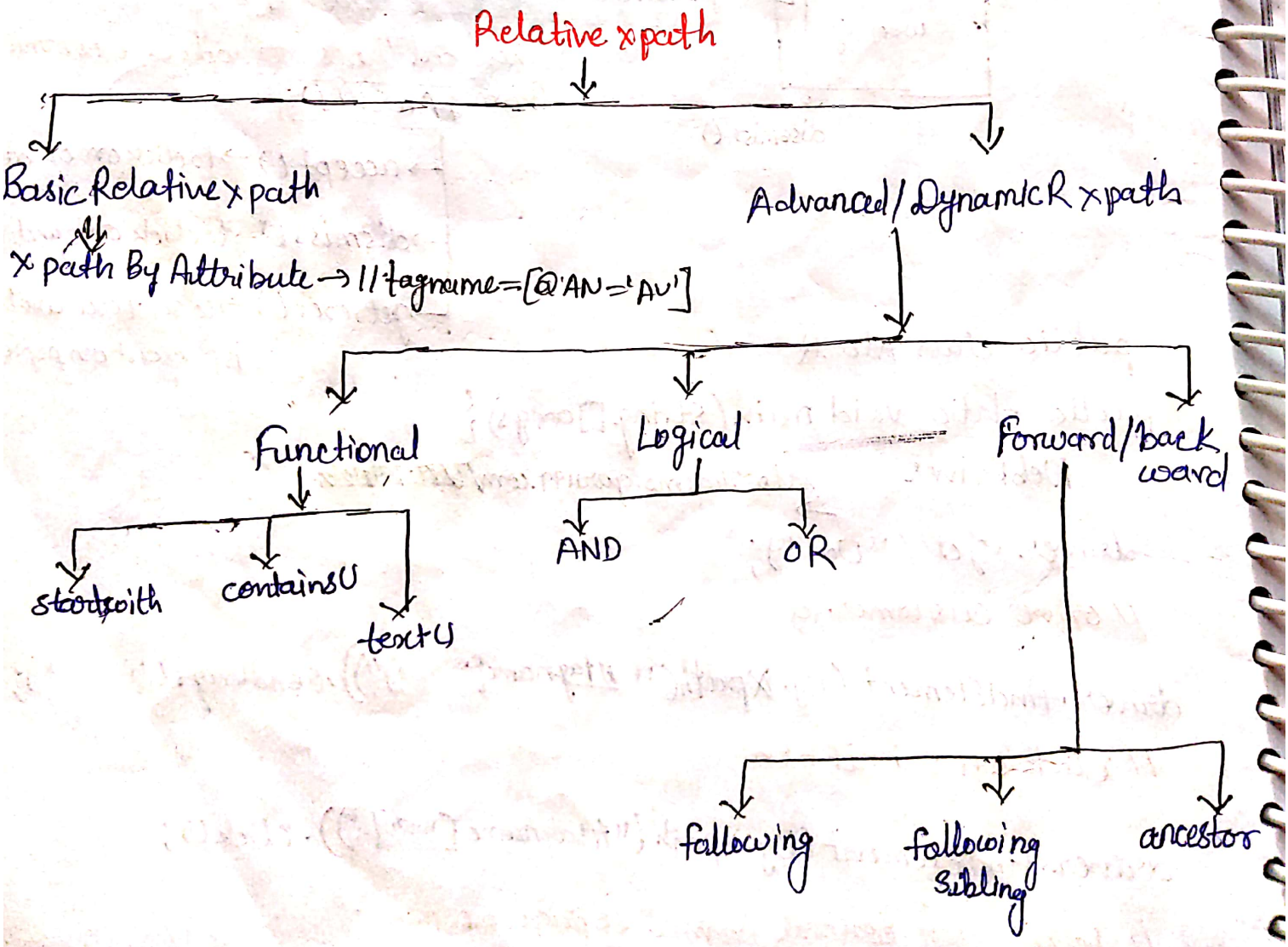
```
driver.switchTo.alert();
```

These all three methods are comming from Alert(I)

- accept() → to click on ok btn
- dismiss() → to click on cancle btn
- getTexet() → to get text which is present on popup

```
public class Alert{
public static void main(String[]args){
    WebDriver driver = https://demo.guru99.com/Test/tabletest
driver.get("Url");
// give customer id
driver.findElement(By.Xpath("//tagname[" "]")).sendkeys(" ");
// click on submit btn.
driver.findElement(By.xpath("//tagname[" "]")).click();
// To switch control from webpage to popup.
Alert alt = driver.switchTo().alert();
```

```
//click on ok btn
alt.accept();
// want for text
alt.getText();
// To print text
System.out.println(alt.getText());
// To click on cancle btn
alt.dismiss();
}
}
```



Functional :-

1. Xpath By starts-with :-

ex → hdfc = (PReddy) 12345678 → starting name is different
= SNaidu 12345679
BReddy 123567 89

Formula :- //tagname[starts-with(@AN, 'AV')]

Program :-

```
public class startswith {
public static void main (String [] args) {
WebDriver driver = new ChromeDriver
driver.get ("URL");
// search for mobiles
driver.findElement (By.xpath (//tagName[starts-with(@AN,'AV')]).
                                                    sendkeys("1")
```

2. Xpath By Contains :-

Program

```
public class contains {
public static void main (String [] args) {
```

3. Xpath By Text :-

//tagName [text() = 'textvalue' / 'visible text']

ex:- //a [text() = 'About']

Program:-

```
public class Text {
public static void main (String [] args) {
 WebDriver driver =new Chromedriver;
 driver.get ("URL")

String t = driver.findElement(By.xpath("xpathexpression")).
                                                    getText();
 system.out.println (t);

 }

}
```

Actions Class:- It is used to handle mouse related actions

1. Move the cursor → moveToElement()
2. Right click action → contextClick()
3. Double click action → doubleClick()
4. Drag And Drop → dragAndDrop()
5. Click action → click()
6. build() ⇒ it is used to connect multiple methods of Actions class in single statement.
7. perform() ⇒ mandatory method we should have to perform the action on element.

ctrl+shift+o to import all

Program for MoveToElement()

```
public class MoveToElement {
public static void main (String[] args){
WebDriver driver = new ChromeDriver();
driver.get ("https://www.google.com");
// xpath for Gmail
WebElement gmail = driver.findElement(By.xpath("//a[text()='Gmail']"));
// Create object of Actions class
Actions a = new Actions (driver);
a.moveToElement (gmail).perform();

}
}
```

⇒ Drag and Drop :-

here we have 2 elements

1) Source element
2) destination element.

Program for Drag & Drop :-

```
public class Drag_drop {
public static void main (String [] args) {
    WD  driver = new ChromeDriver();
driver.get ("https://www.w3schools.com");
 // Src element
WebElement  srcele = driver.findElement(By.xpath("—"));
// dest element
WebElement destele = driver.findElement(By.xpath("—"));
// Create object of Actions class
Actions  a = new Actions (driver);
  a.dragAndDrop(Srcele, destele).perform();
}
}
```

Child Browser Popup / window Handling [27/09/24]

Characterstics :- ① We can inspect element present on popup.
② It will contains address field (Url), maximize, minimize, and close option.
③ Each and every window will generate the session ID

How to get all windows ? ⟹ getwindowHandles()

Returntype of this methods is " Set<String> "
↳ Interface which not allows duplicates

Syntax:-
```
Set<String> allwindow = driver.getwindowHandles();
for (String window : allwindow) {
    driver.switchTo().window(window);
}
```

⟹ How to Handle windows in selenium?

⟹ By default control of selenium is on main webpage, we should have to switch the control from main webpage to particular window

```
driver.switchTo.window();
```

Script for handling window :-

```
public class MWH {
public static void main (String[] args) {
WebDriver driver = new ChromeDriver();
driver.get ("https://www.flipkart.com");
```

// ~~click on~~ search for mobiles under 20k

driver.findElement(By.xpath("__")).sendKeys("mobiles under 20k");

// click on search btn

driver.findElement(By.xpath("__")).click();

// click on 1st phone

driver.findElement(By.xpath("__")).click();

// check where is control of selenium

String exmpt="__";

String mwt = driver.getTitle();

if (mwt.equals(exmpt)) {

System.out.println("still driver is on main webpage");

}

else {

System.out.println("driver is navigated to child window");

}

//How To switch

Set<String> allwin = driver.getWindowHandles();

ArrayList<String> al = new ArrayList<String>(allwin);

System.out.println(al.get(0)); //print session ID of main win

System.out.println(al.get(1)); //print session ID of child windo

//switch to child

driver.switchTo().window(al.get(1));

}

}

[28/9/24]

Keysclass/Keboard stroke:-

"Pressing any keys on Keyboard" Handling scenarios of keyboard is called as Keyboard stroke.

using this class; we can reduce the length of script

- It is used to automate the keys which are present on Keyword - Mostly keys class can be used in sendkeys method

Drawback:-

It won't automate special characters.

Ex. How to use keyclass

```
public class usingKeyclass {
public static void main (String[] args) {
WebDriver driver = ChromeDriver();
driver.get ("URL");
driver.findElement(By.xpath("username")).SendKeys("abc",Keys.TAB,"xyz@123",
                                                  .Keys.ENTER);

}

}
```

Robot class → It will remove drawbacks of keys class "or"

To avoid drawback of keys class we have Robot class. It is coming from "Java.awt package

Syntax:-

```
Robot r = new Robot();
r.keyPress(KeyEvent.virtualKey.PAGE_Down);
r.KeyRelease(KeyEvent.virtualKey.PAGE_Down);
}
}
```

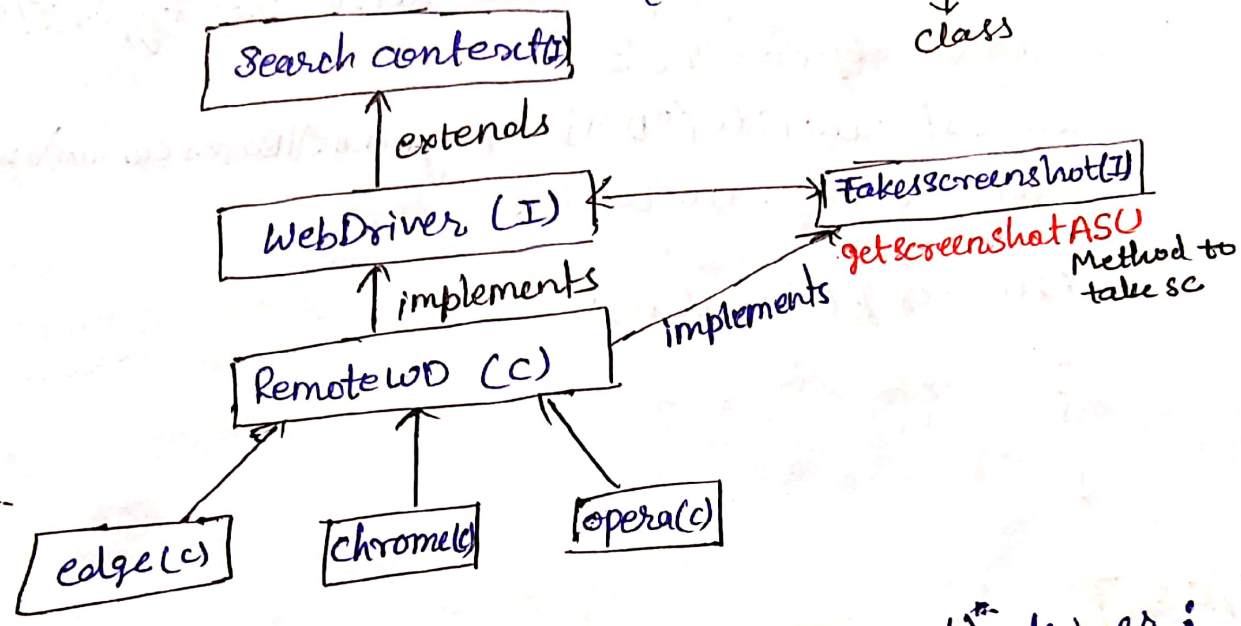
Ex

```
public class Robotclass {
public static void main(String[]args){
WebDriver driver = new ChromeDriver();
driver.get("URL");
//create object for Robotclass
Robot r = new Robot();
r.keyPress(KeyEvent.VK_PAGE_DOWN);
r.KeyPress(KeyEvent.VK_PAGE_DOWN);
Thread.sleep(2000);
```

Screenshot :— It is phototype without any functionality.

- first of all we have to type cast TakesScreenshot(I).

Inside this method getScreenshotAs() method is there to take screenshot. It takes argument (OUTPUTTYPE.FILE) → Interface and we have to store this screenshot in File and we have to store in Desktop. Then we have to call the method File.copy(src, dest); ↓ class



TakesScreenshot ts = (TakesScreenshot) driver; ↳ Typecasting operator

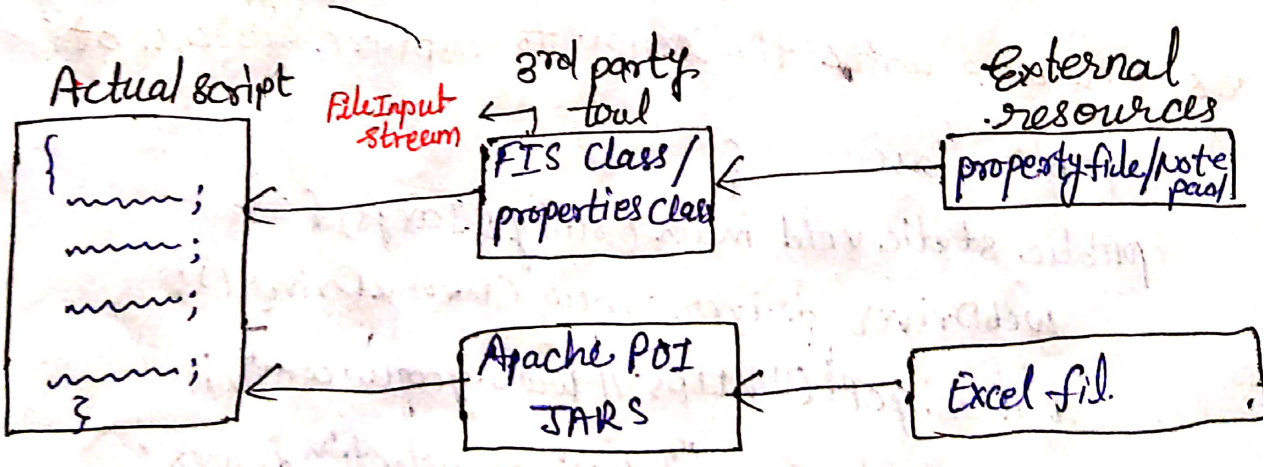
Ex :- How to write the script to capture screenshot :?

```
public class {
public static void main (String[] args) {
    WebDriver driver = new ChromeDriver();
    driver.get ("https://www.google.com");
TakesScreenshot ts = (TakesScreenshot) driver;
    File src = ts.getScreenshotAs(OUTPUTTYPE.FILE);
    File dest = new File("Path of folder\\scname.ext)
    Files.copy (src, dest);
}
```

```
public class IndivisualSc {
public static void main (String [] args) {
  WebDriver driver = new ChromeDriver();
  driver.get ("https://www.google.com");
  // write locator for logo
   WebElement logo = driver.findElement(By.xpath("—"));
      File src = logo.getScreenshotAs(OutputType.FILE);
   // create object file class
   File dest= new File ("D:\\java programms\\Basics selenium\\SCREENSHOT");
   // copy sc from src to dest

    File.copy (src, dest);

   }
   }
```

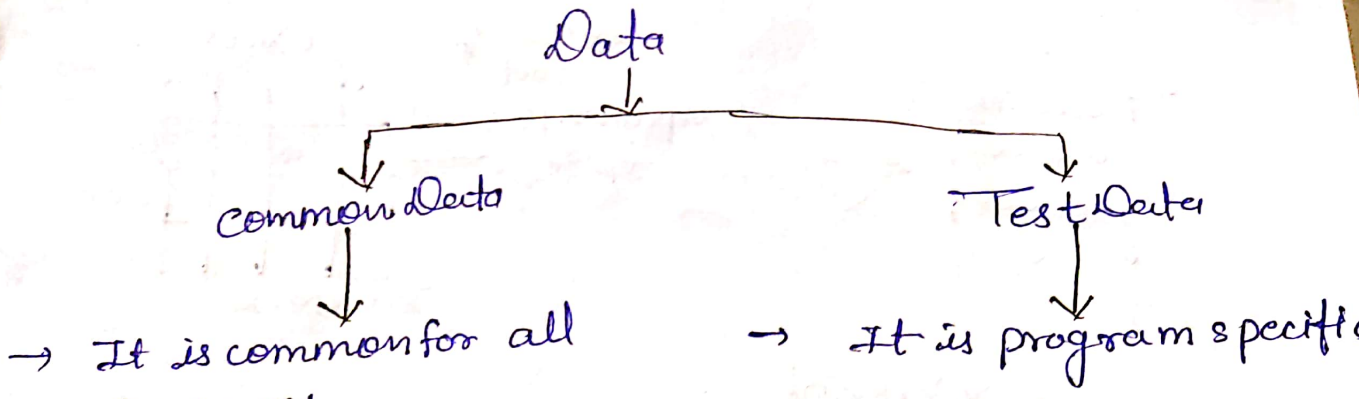
[01/10/24]

Data Driven Testing! –



⇒ The process of fetching data from external resources using 3rd party tools is called as "Data Driven Testing!"

there are two types of Data in testing



→ It is common for all programs

→ we can maintain it in .properties files (Notepad)

ex- Url, userName, password, browsers

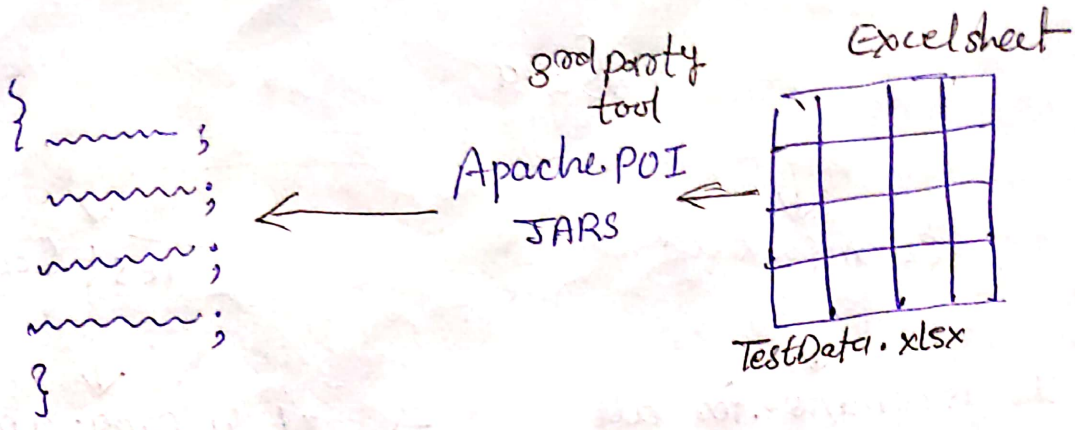
→ It is program specific

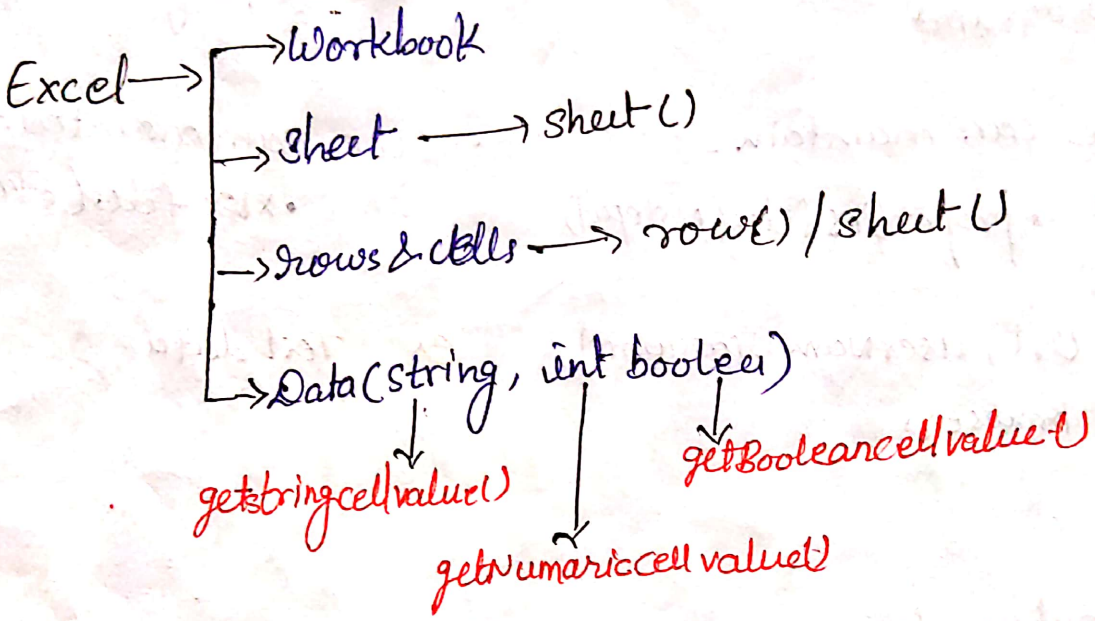
→ we can maintain it in .xlsx file (Excel)

ex Test data

```
public class NotepadDF{
public static void main(String[] args){
// obj of FIS class
FileInputStream fisp = new FileInputStream("path of .property file");
Properties p = properties();
p.load(fisp);
String UN = p.getproperty("un");
System.out.println(UN); // admin
}
}
```

Excel Data Fetching :- [08/10/24]





⇒ To open Excel sheet we will use ".create()" method. which comes from "workbook factory class".

Syntax ~~FIS fise~~.

```
FileInputStream fise = new FIS("pathof ExcelFile");
workbook wb = WorkbookFactory.create(fise);
String data1 = wb.sheet("Sheet1").getrow(1).cell( ).
                                  getstringcellvalue();
System.out.println(data1);
}
```

(rowvalue, cellvalue)

How to download Apache POI JARS?

Chrome → download Apache POI JARS → Binary → 5.2.3 .Zip

[03/10/24]

Domain - Manufacturing Domain

Project Name - V tiger [CRM /Scalar, Quantas CRM

Login page :-

[04/10/24]

TestNG :- (test Next Generation)

⇒ It is the Basic framework used to automate the manual Test cases

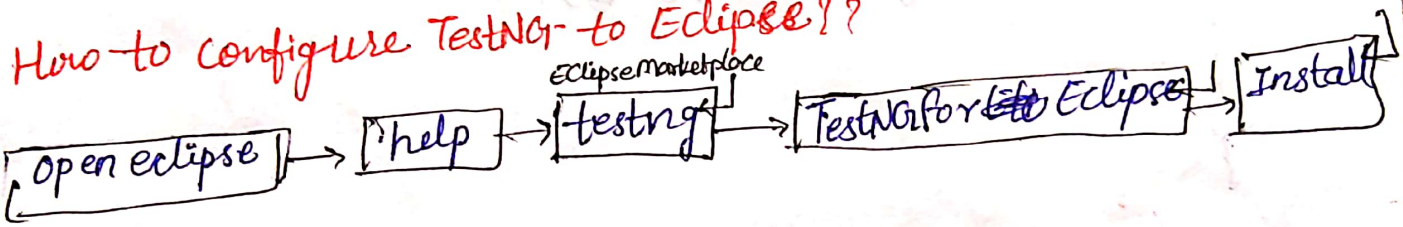
Characterstics :-

It contains @Test [TestAnnotation]

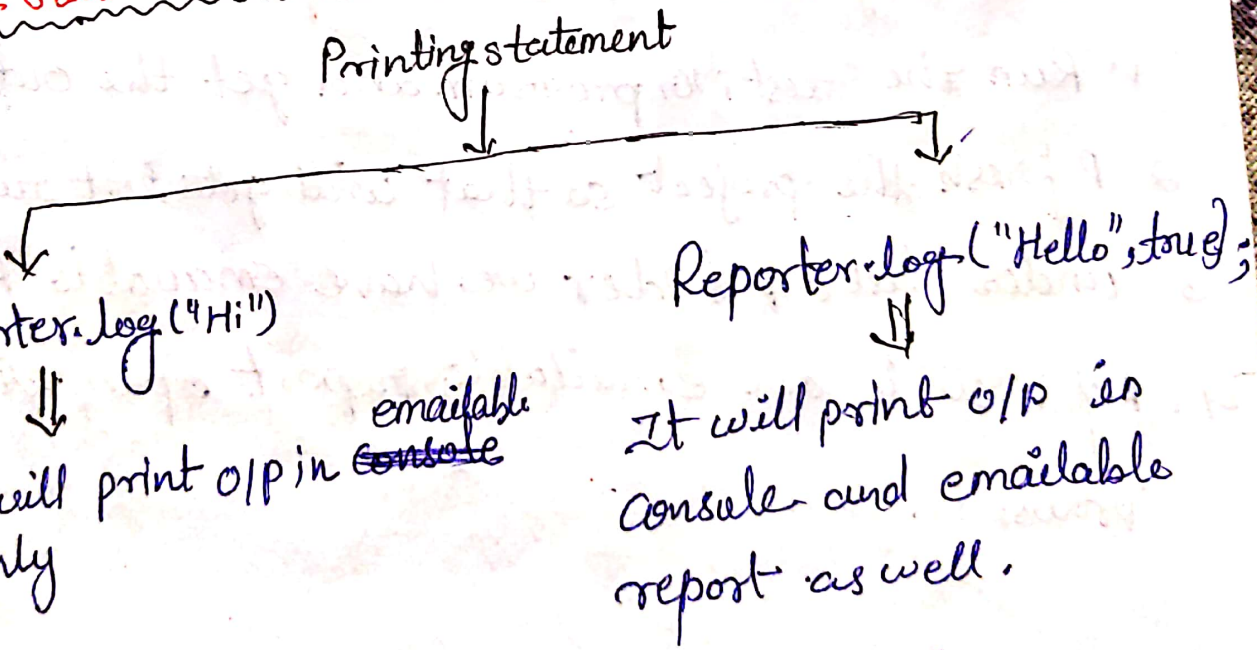
(act as main method) (act as Test case)

NOTE - each @Test followed by non-static method.

⇒ How to configure TestNG to Eclipse??



* Printing statement in TestNG



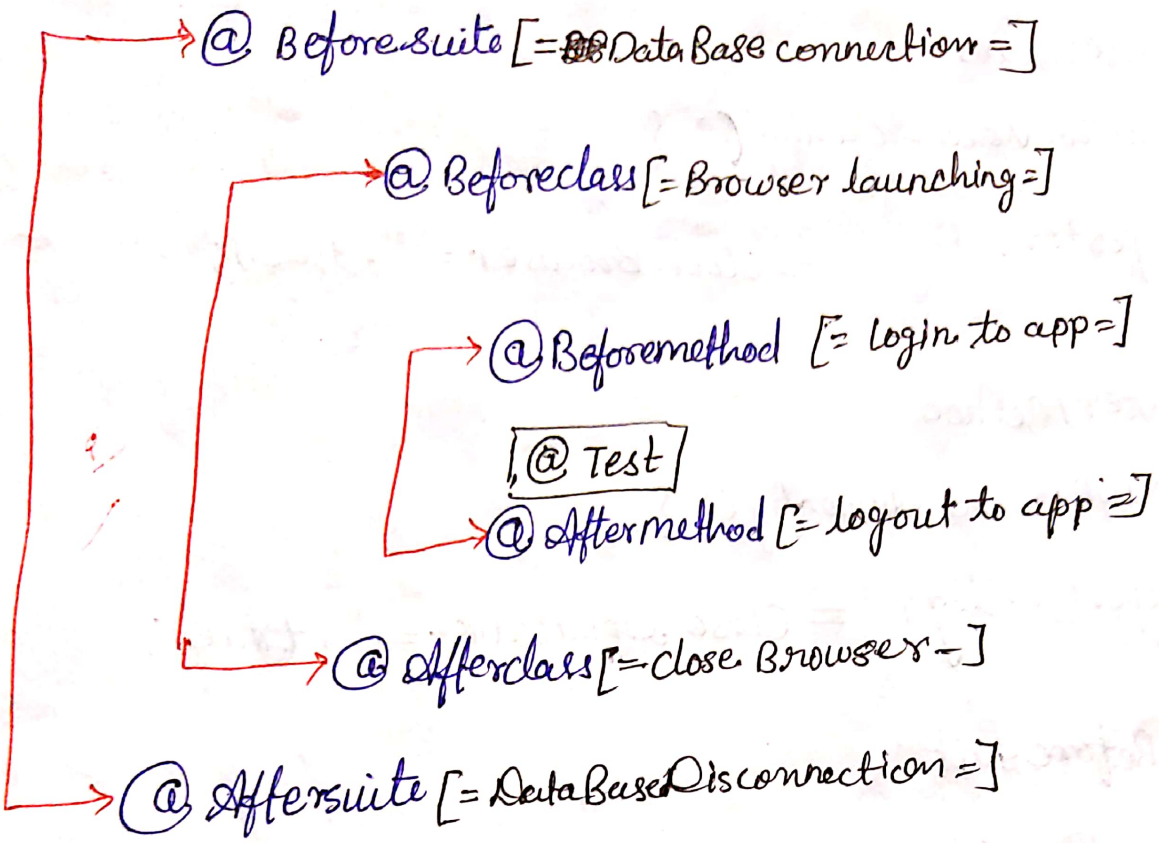
```
public class printing {

@Test [Test annotation]
public void TC1() {
  Reporter.log ("Hgjf"); // it will not print output in Consolo
  }
@Test [Test annotation]
public void TC2() {
   Reporter.log ("Hello", true);
  }
}
```

⇒ How to generate Emailable Report ?

1. Run the testNG program and get the output
2. Refresh the project so that will get test output folder
3. under test o/p folder we have emailable Report
1. Right click on emailable report open with web browser.

TestNG Annotation

Annotations are inbuilt codes which can perform specific task.



program :-

```
public class Annotation_flow {
@ Test
public void tc() {
Reporter.log("=Actual Tc=", true);
}
@ Before class
 public void bcconfig() {
Reporter.log("= launch Browser=", true);
}
@ Before Method
   public void bmconfig() {
```

```
Reporter.log("=Launch Application=", true);
}
@AfterClass
public void Aconfig() {
Reporter.log("=Close Browser=", true);
}
@AfterMethod
public void Mconfig() {
Reporter.log("=Close Application=", true);
}
@BeforeSuit
public void bsconfig() {
Reporter.log("=DB connection=", true);
}
@AfterSuit
public void asconfig() {
Reporter.log("=DB Disconnection=", true);
}
}
```

[7/10/24]

TestNG-Keywords / TestNG Flags

Mainly we have 5 keywords, Keywords always starts with lower case

1. invocation count
2. timeout
3. enable = false
4. dependson methods
5. priority

} Code optimization

⇒ **invocation count :-** If we want to run same TestCase multiple times then we can use it.

ex :-

```
@Test (invocationCount = 100)
public void m1() {
Reporter.log("Hi", true);
}
```

[Selenium (CrossBrowser / compatibility)]

⇒ **timeout :-** If one of the TC is taking to much time for Execution then we give specific time using timeouts keywords.

ex :-

```
@Test (
public void m1() {
Reporter.log(
```

⇒ enabled = false :- It will Ignore the particular TC

```
@Test
public void m1()
```

⇒ dependsonMethod

Assert class / Hard Assert :- Till now, for verification we have used if, else statement But now we are using Assert class [Assertions].

Mostly we will use :- ① Assert.assertEquals
② Assert.assertNotEquals
③ Assert.fail()
↓
when intentionally we want to fail TC we will use Assert.fail()

=# Assert.assertEquals :- Whenever we want to check 2 strings are equal or not.

Ex :-

```
class A {
@Test
public void m1() {
    String s1 = "Hi";
    String s2 = "Hello";
    Asser.assertEquals(s1, s2); //fail
}
}
```

Assert.assertNotEquals :- It is used to compare strings whenever two strings are not same then o/p is pass otherwise fail

```
public class B {
@Test
public void m2() {
String s1 = "Hi";
String s2 = "Hello";
Assert.assertNotEquals(s1, s2); //pass
}
}
```

Assert.fail() :-

```
public class c {
@Test
public void m1() {
system.out
```

Difference b/w enabled = false & Assert.fail()

⇒ enabled = false ignores Test case	⇒ Assert.fail fails TC intentionally
⇒ we can write it infront ~~after~~	→ we can not write it ~~at~~ infront of @Test
@Test(enabled=false)	@Test(Asser.fail) ✗

[9/10/24]

Difference between Assert & verify :- (Vimp)

Assert	Verify
→ It one of the verifications belongs to Tc is failed.	→ It one can on verification is belongs to Tc thah will execute.
→ All methods are static Ex:- Assert.assertEquals() Assert.fail;	→ All methods are non static.
→ No need of object creation because methods are static.	→ object creation is mandatory.
→ assertAll() is not mandatory	→ assertAll() is mandatory.
→ It comes under Hard Assert	→ It comes under soft Assert.
→ In Hard Assert, If verification 1 is failed then it stops the execution.	→ In soft Assert, if verification 1 is failed then next verification will be executed.

Batch Execution :-

The process of running multiple classes with a single click or using single xml file is called as "Batch Execution."

Ex:-

(class) Sample 1	(class) Sample 2	(class) Sample 3
3 { TC1, TC2, TC3	3 { TC4, TC5, TC6	3 { TC7, TC8, TC8

⇒ Total 9 TC's. All 9 TC's belong to 3 different classes we can run all 3 classes at a time. But all the classes must be long to same package.

Program 1:-

```
Package Batch_Execution;
import org.testing.Reporter;
import org.testing.annotation.Test;

public class Sample1
{
@Test
public void TC1(){
Reporter.log("TC1;true);
}
@Test
public void TC2(){
Reporter.log("TC2", true);
}
@Test
public void TC3(){
Reporter.log("TC3", true);
}
}
```

Program 2 :-

```
package Batch_Execution;
import org.testng.Reporter;
import org.testing.annotations.Test;

public class Sample2 {
@Test
public void TC4() {
Reporter.log("TC4", true);
}
@Test
public void TC5() {
Reporter.log("TC5", true);
}
@Test
public void TC6 {
Reporter.log("TC6", true);
}
}
```

Program 3 :-

```
Package Batch_Execution;
import org.testing.Reporter;
import org.testing.annotations.Test;
public class Sample3 {
@Test
public void TC7() {
Reporter.log("TC7", true);
@Test
public void TC8() {
Reporter.log("TC8", true);
}
@Test-
```

```
public void TC9(){
Reporter.log("TC9", true);
```

Procedure for Batch Execution:-

1) Select 3 classes [hold control Button & select it]
2) Right click on 3 classes → TestNG → convert TestNG
3) We will get XML file i.e Sample1, Sample2 & Sample3 then run the suite.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testing.org/testng-1.0.dtd">
    <suite name="Suite">
<test thread-count="5" name="Test">
  <classes>
<class name="Batch_Execution.Sample1" />
<class name="Batch_Execution.Sample2" />
<class name="Batch_Execution.Sample3" />
  </classes>
</test> <!--- Test -->
</suite> <!--- Suite -->
```

⇒ How to ignore TC through Suite?

Ex.- Sample TC1, TC2, TC3
We want to ignore TC2

```
Suite:- <classes>
  <class name="TestNG.Sample">
  <methods>
<exclude name="TC2" /> </exclude>
```

</methods>

</class>

</classes>

⇒ How we can disable the TC ?

A. There are 2 ways to disable.

1) using class level → using enabled = false.

2) using Suite level → using exclude keyword.

Program:- Package DisableTC;

```
import org.testng.Reporter;
import org.testng.annotations.Test;

Public class Demo{
@Test
public void TC1() {
Reporter.log ("TC1",true);
}
@Test
public void TC2(){
Reporter.log ("TC2",true);
}
@Test
public void TC3(){
Reporter.log ("TC3", true);
}
}
```

→ Then right click on Demo class, then testNG, then convert to TestNG. Then give the name as Disable TC.XML. Then click & open Diable TC.XML.(Pass)

```
<?xml version = "1.0" encoding = "UTF-8"?>
<!DOCTYPE Suite SYSTEM "https://testng.org/testng-1.0.dtd">
<Suite name = "Suite">
<test Thread-count = "5" name = "Test">
 <classes>
 <class name = "Disable Tc.Demo">
   <methods>
 <exclude name = "TC2"></exclude>
 </methods>
 </class>
 </classes>
 </test><!---Test-->
 </suite><!Suite-->
```

Parallel Testing / Parallel Execution :-

It is the process of running multiple test cases parallely at a time.

Ex :- Amazon & Flipkart ~~.com .org~~ on one browsers at a time.

Program to open Amazon :-

```
Package Parellel-testing;

import org.openqa.Selenium.webDriver;
import org.openqa.Selenium.ChromeDriver;
import.org.testng.annotations.Test;
```

```
Public class A {
@Test
public void TC1L){
WebDriver driver = new ChromeDriver();
driver.get (https://www.amazon.in");
}
}
```

Program open to flipkart :-

```
Package Parallel Testing;
import org.openqa.Selenium.WebDriver;
import org.openqa.selenium.ChromeDriver;
import org.testng.annotations.Test;
public class B {
@Test
public void TC 1L){
WebDriver driver = new ChromeDriver();
driver.get ("https://www.flipkart.com);
}
}
```

Procedure for parallel suite :-

1) Copy from test to test i.e as follows. Go to xml file.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="Suite" parallel="tests">
<test thread count="5" name="Test1">
<classes>
```

17

use of @findBy:- It will findElement even after refreshing the page on session ID

use of findElements:- It won't capture updated session ID on a page gets refreshed. findElement method work older session ID.

⇒ Who is helping to @FindBy to capture updated session ID?

⇒ initElements() from page Factory class.

```
Class loginpage{
// Declaration
@FindBy(xpath="___") private web untext;

// initialization
public Loginpage(WebDriver driver){
PageFactory.initElements(driver, this);
```

// utilization (Create getter)

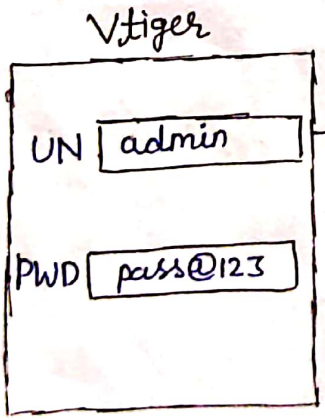
Right click → source → generate getters & setters → select only getter → Finish.

```
package objectRepos;

public class LoginPage {

// Declaration
@FindBy (xpath = "") private WebElement untxt;   // xpath Expression
@FindBy (xpath = "") private WebElement pwdtxt;
@FindBy (xpath = "") private WebElement loginBtn;

// Initialization
  public Loginpage (WebDriver driver) {
    pageFactory.initElements (driver, this);
}
// Utilization
public WebElement getUntxt() {
    return untxt;
}
public WebElement getPwdtxt() {
    return pwdtxt;
}
public WebElement getLoginBtn() {
    return loginBtn;
  }
}
```

```
<class name = "Parallel testing .A" />
        </classes>
</test><!---Test --->
<test thread-count = "5" name = "Test 2">
        <classes>
<class name = "parallel testing .B" />
        </classes>
</test> <!--Test-->
</suite><!--suite-->
```

[14/10/24]

Data provider :- Through the same program we can give data using Data provider Annotation.



→ Multidimentional Array will use.

```
public class Dataprovider {
@Test
public void m1 (String u, string P) {
WebDriver driver = new ChromeDriver();
driver.get ("Url");
// login
```

```
driver.findElement(By.xpath("—")).sendkeys(u);
driver.findElement(By.xpath("—")).sendkeys(p);
driver.findElement(By.xpath("—")).click();
}
@Dataprovider
public string[][] login(){
String[][] s = new string[2][2];
a[0][0] = "admin";
[0][1] = "pass@123"
[1][0] = "ADMIN"
[1][1] = "PASS@123"
return s;
}
}
```

multidimentional [][] is for represent matrix of value

[15/10/24]

Framework:- These are set of Rules and Regulations which can be followed by Automation Test Engineer. Without framework the code might merge or there is no uniformality over the employee. There is no code optimization. If we want to change Test data / common data, then we have to change for all script [without Fw we can hardcode the data]

Using Framework:- Fw produces beneficial outcomes for user i.e it seprates code & Test data from eachother and if we want to make any changes in the data it can't disturb the script.

★ Due to FW uniformity reflects among all employees

★ It helps for code reusibility

★ It helps for code optimization.

Q: How to create Hybrid FW ?

⇒ We have to create Maven project Hybrid